Logică pentru informatică - Săptămâna 8
Sintaxa logicii de ordinul I

Ștefan Ciobâcă

December 10, 2017

1 Motivație și introducere

Logica de ordinul I, pe care o vom studia în continuare, este o extensie a logicii propoziționale, extensie care aduce un plus de expresivitate. Expresivitatea adițională este necesară pentru a putea modela anumite afirmații care nu pot fi exprimate în logica propozițională, dar care apar în practică.

În logica propozițională, nu putem exprima într-un mod natural următoare afirmație: *Orice om este muritor*.

Pentru a modela o afirmație în logica propozițională, identificăm propozițiile atomice. Apoi asociem fiecărei propoziții atomice o variabilă propozițională. Propozițiile atomice sunt propozițiile care nu pot fi împărțite în una sau mai multe propoziții mai mici, care să fie conectate între ele prin conectorii logici din logica propozițională: *și*, *sau*, *non*, *implică* și respectiv *dacă și numai dacă*.

Observăm că afirmația *Orice om este muritor* nu poate fi descompusă în afirmații mai mici legate între ele prin conectorii logicii propoziționale, după cum este descris mai sus. Așadar, în logica propozițională, afirmația este atomică. Asociem întregii afirmații o variabilă propozițională $\mathtt{p} \in A$.

Acum să modelăm afirmația *Socrate este om.* Evident, acestei a doua afirmații trebuie să îi asociem o altă variabilă propozițională $\mathtt{q} \in A$. Să presupunem că știm că $\mathtt{p}$ și $\mathtt{q}$ sunt adevărate. Formal, știm că lucrăm cu o interpretare $S : A \to B$ astfel încât $S(\mathtt{p}) = 1$ și $S(\mathtt{q}) = 1$. Putem trage concluzia ca afirmația *Socrate este muritor* este adevărată în interpretarea S?

Nu, deoarece afirmației *Socrate este muritor* ar trebui să îi asociem o a treia variabilă propozițională $\mathtt{r}$ și nu putem trage nicio concluzie asupra lui $S(\mathtt{r})$ din faptul că $S(\mathtt{p}) = 1$ și $S(\mathtt{q}) = 1$. Deci, din semantica logicii propoziționale, nu putem trage concluzia ca $\mathtt{r}$ este adevărată în orice interpretare în care $\mathtt{p}$ și $\mathtt{q}$ sunt adevărate, în ciuda faptului că, în orice lume în care orice om este muritor și Socrate este om putem trage concluzia că Socrate este muritor. Această diferență între realitate și modelarea noastră ne indică faptul că modelarea nu este suficient de bună.

Logica de ordinul I aduce, în plus față de `LP`, noțiunea de *cuantificator* (existențial sau universal) și noțiunea de *predicat*. Cuantificatorul universal este notat cu $\forall$, iar cuantificatorul existențial este notat cu $\exists$.

Un predicat este o afirmație a cărei valoare de adevăr depinde de zero sau mai mulți parametri. De exemplu, pentru afirmație de mai sus, vom folosi două predicate: *Om* și *Muritor*. Predicatul *Om* va fi predicatul care denotă umanitatea: $Om(x)$ va fi adevărat când x este om. Predicatul *Muritor* este adevărat când argumentul său este muritor. Deoarece predicatele de mai sus au fiecare câte un singur argument/parametru, ele se numesc predicate *unare*. Predicatele generalizează variabilele propoziționale prin faptul că pot primi argumente.

Astfel, afirmația *orice om este muritor* va fi modelată prin formula

$$\forall x.(Om(x) \rightarrow Muritor(x)),$$

care este citită astfel: *pentru orice x, dacă Om de x, atunci Muritor de x.* Afirmația *Socrate este om* va fi modelată prin formula $Om(s)$, unde s este o *constantă* prin care înțelegem Socrate, la fel cum prin constanta 0 ne referim la numărul natural *zero*. De exemplu, $Om(s)$ este adevărat (deoarece s denotă un om), dar $Om(l)$ este fals dacă l este o constantă care ține locul cățelului *Lăbuș*.

Afirmația *Socrate este muritor* va fi reprezentată prin $Muritor(s)$ (țineți minte că constanta s se referă la Socrate). Afirmația $Muritor(s)$ este adevărată deoarece Socrate este muritor; la fel și afirmația $Muritor(l)$ este adevărată.

Vom vedea că în logica de ordinul I, formula $Muritor(s)$ este consecință a formulelor $\forall x.(Om(x) \rightarrow Muritor(x))$ și respectiv $Om(s)$. În acest sens, logica de ordinul I este suficient de expresivă pentru a explica din punct de vedere teoretic raționamentul prin care putem deduce că *Socrate este muritor* din faptul că *Orice om este muritor* și din faptul că *Socrate este om*.

2 Structuri și signaturi

Cu siguranță ați întâlnit deja mai multe formule din logica de ordinul I, fără să știți neapărat că aveți de a face cu logica de ordinul I. Fie următoare formulă:

$$\varphi = \forall x.\forall y.\Big(x < y \rightarrow \exists z.(x < z \land z < y)\Big).$$

Formula folosește un predicat binar, $<$, care este definit astfel: $<(x, y)$ este adevărat dacă x este mai mic strict decât y. Pentru multe predicate binare (inclusiv pentru $<$), pentru a simplifica scrierea, folosim notația infixată ($x < y$) în loc de notația prefixată ($<(x, y)$).

Este formula φ de mai sus adevărată? Formula afirmă că între orice două valori ale variabilelor x, y există o a treia valoare, a variabilei z. Formula este adevărată dacă domeniul variabilelor x, y, z este $\mathbb{R}$, dar este falsă dacă domeniul este $\mathbb{N}$ (între orice două numere reale există un al treilea, dar între două numere naturale consecutive nu există niciun alt număr natural).

În general, formulele de ordinul I se referă la o anumită *structură matematică*.

Definition 2.1. *O* structură matematică *este un tuplu*

$$(D, R_1, \ldots, R_n, f_1, \ldots, f_m),$$

unde:

- *D este o mulțime nevidă numită* domeniu*;*
- *R_i ($1 \leq i \leq n$) sunt predicate (de o aritate oarecare) peste mulțimea D;*
- *f_i ($1 \leq i \leq m$) sunt funcții (de o aritate oarecare) peste mulțimea D.*

Funcții și predicate peste o mulțime D

Definition 2.2. *O* funcție de aritate n peste mulțimea D *este orice funcție*

$$f : D^n \to D.$$

Definition 2.3. *Un* predicat de aritate n peste mulțimea D *este orice funcție*

$$p : D^n \to B,$$

unde $B = \{0, 1\}$ este mulțimea valorilor de adevăr.

Iată câteva exemple de structuri matematice:

1. $(\mathbb{N}, <, =, +, 0, 1)$;

   Domeniul structurii este mulțimea numerelor naturale. Structura conține două predicate: $<$ și $=$, ambele de aritate 2. Predicatul $<$ este predicatul *mai mic* pe numere naturale, iar predicatul $=$ este predicatul de *egalitate* a numerelor naturale.

   De asemenea, structura conține trei funcții. Funcția binară $+ : \mathbb{N}^2 \to \mathbb{N}$ este funcția de adunare a numerelor naturale, iar funcțiile $0 : \mathbb{N}^0 \to \mathbb{N}$ și respectiv $1 : \mathbb{N}^0 \to \mathbb{N}$ sunt funcțiile de aritate 0 (numite funcții constante sau pur și simplu constante) 0 și 1.

   Funcțiile de aritate 0

   O funcție de aritate 0 se mai numește și *constantă*. Conform definiției noastre, o astfel de funcție este de forma

   $$c : D^0 \to D.$$

   În general, mulțimea D^n este produsul cartezian al mulțimii D cu ea însăși de n ori. Formal, avem că

   $$D^n = \{(d_1, \ldots, d_n) \mid d_1 \in D, \ldots, d_n \in D\}.$$

   În cazul particular în care $n = 0$, mulțimea D^0 are un singur element, și anume tuplul vid. Din acest motiv, funcțiile de aritate 0 nu primesc niciun argument și întorc tot timpul același rezultat. Practic, pentru orice constantă $c \in D$, există o unică funcție $: D^0 \to D$ care să întoarcă c. Nu vom face nicio diferență între c și această funcție și din acest motiv vom nota funcția tot cu c.

2. $(\mathbb{R}, <, =, +, -, 0, 1)$;

   Această structură conține două predicate binare, $<$ și $=$, precum și trei funcții peste $\mathbb{R}$: funcția binară $+$, funcția unară $-$ și constantele $0, 1 \in \mathbb{R}$.

3. $(\mathbb{Z}, <, =, +, -, 0, 1)$;

   Această structură este similară cu cea de mai sus, dar domeniul este mulțimea numerelor întregi.

4. $(B, \cdot, +, \text{-})$;

   Această structură este o algebră booleană, unde domeniul este mulțimea valorilor de adevăr, iar funcțiile sunt cele cunoscute din prima jumătate a semestrului. Astfel de structuri, fără niciun predicat, se numesc *structuri algebrice*.

5. $(\mathbb{R}, <)$.

   Această structură conține doar un predicat de aritate 2 (relația *mai mic* peste $\mathbb{R}$) și nicio funcție. Structurile care nu conțin funcții se numesc structuri relaționale. Structurile relaționale cu domeniul finit se mai numesc baze de date relaționale și se studiază in anul 2.

Revenind la formula de mai devreme:

$$\varphi = \forall x.\forall y.\Big(x < y \rightarrow \exists z.(x < z \land z < y)\Big),$$

avem că această formula este adevărată în structura $(\mathbb{R}, <, =, +, -, 0, 1)$ (între orice două numere reale distincte există cel puțin un număr real) dar este falsă în structura $(\mathbb{Z}, <, =, +, -, 0, 1)$ (deoarece nu între orice două numere întregi putem găsi un alt număr întreg – de exemplu între două numere întregi consecutive nu există niciun întreg).

Când avem o formulă de ordinul I și dorim să îi evaluăm valoarea de adevăr, trebuie să fixăm structura în care lucrăm.

Este posibil ca două structuri diferite să aibă un set de predicate și de funcții cu același nume. De exemplu, chiar structurile de mai devreme, $(\mathbb{R}, <, =, +, -, 0, 1)$ și respectiv $(\mathbb{Z}, <, =, +, -, 0, 1)$. Deși predicatul $<: \mathbb{R}^2 \rightarrow \mathbb{R}$ este diferit de predicatul $<: \mathbb{Z}^2 \rightarrow \mathbb{Z}$, ele au același nume: $<$.

În general, în Matematică și în Informatică, nu facem diferența între un predicat și numele lui, respectiv între o funcție și numele funcției, dar în Logică diferența este extrem de importantă. În particular, dacă ne referim la numele unei funcții vom folosi sintagma "simbol funcțional", iar dacă ne referim la numele unui predicat vom folosi sintagma "simbol predictiv". De ce este importantă diferența între un simbol predicativ și un predicat? Deoarece vom avea (ne)voie să asociem simbolului predicativ diverse predicate, analog modului în care unei variabile într-un limbaj de programare imperativ îi putem asocia diverse valori.

Când ne interesează doar numele funcțiilor și predicatelor (nu și funcțiile și respectiv predicatele în sine), vom lucra cu signaturi:

Definition 2.4. *O* signatură Σ *este un tuplu* $\Sigma = (R_1, \ldots, R_n, f_1, \ldots, f_m)$ *unde* $R_1, \ldots, R_n$ *sunt simboluri predicative și* $f_1, \ldots, f_m$ *sunt simboluri funcționale, cu proprietatea că fiecare simbol predicativ* R_i *(*$1 \leq i \leq n$*) și fiecare simbol funcțional* f_i *(*$1 \leq i \leq m$*) are asociat un număr natural numit* aritatea *simbolului. Vom nota cu* $ar(f_i)$ *aritatea simbolului funcțional* f_i *(*$1 \leq i \leq m$*) și cu* $ar(R_i)$ *aritatea simbolului predicativ* R_i *(*$1 \leq i \leq n$*).*

Unei signaturi îi putem asocia mai multe structuri:

Definition 2.5. *Dacă* $\Sigma = (R_1, \ldots, R_n, f_1, \ldots, f_m)$ *este o signatură, o* Σ-structură *este orice structură* $S = (D, R_1^S, \ldots, R_n^S, f_1^S, \ldots, f_m^S)$ *astfel încât* R_i^S *este un predicat peste* D *de aritate* $ar(R_i)$*, iar* f_i^S *este o funcție peste* D *de aritate* $ar(f_i)$*.*

De reținut!
Structură = domeniu + predicate + funcții
Signatură = simboluri predicative + simboluri funcționale
Unei signaturi Σ îi putem asocia mai multe structuri, numite Σ-structuri.

Example 2.1. *Fie* $\Sigma = (P, Q, f, i, a, b)$ *unde* P, Q *sunt simboluri predicative de aritate* $ar(P) = ar(Q) = 2$ *și* f, i, a, b *sunt simboluri funcționale cu aritățile:* $ar(f) = 2$*,* $ar(i) = 1$ *și* $ar(a) = ar(b) = 0$*.*

Avem că $(\mathbb{R}, <, =, +, -, 0, 1)$ *și respectiv* $(\mathbb{Z}, <, =, +, -, 0, 1)$ *sunt* Σ*-structuri.*

Dacă am fixat o signatură $\Sigma = (R_1, \ldots, R_n, f_1, \ldots, f_m)$, notăm cu $\mathcal{P} = \{R_1, \ldots, R_n\}$ mulțimea simbolurilor predicative din signatură, iar cu $\mathcal{F} = \{f_1, \ldots, f_m\}$ mulțimea simbolurilor funcționale din signatură. Mulțimea simbolurilor predicative de aritate n este notată cu $\mathcal{P}_n = \{P \mid ar(P) = n\}$, iar mulțimea simbolurilor funcționale de aritate n este notată cu $\mathcal{F}_n = \{f \mid ar(f) = n\}$.

3 Sintaxa logicii de ordinul I

În continuare, vom studia sintaxa (definiția matematică a modului în care se scriu) formulelor din logica cu predicate de ordinul I și semantica acesteia (cum calculăm valoarea de adevăr a unei formule).

Spre deosebire de logica propozițională, unde limbajul (mulțimea de șiruri de caractere) `LP` este unic determinat, există mai multe limbaje de ordinul I, câte un limbaj pentru fiecare signatură Σ.

În continuare, vom presupune fixată o signatură Σ cu simboluri predicative $\mathcal{P}$ și simboluri funcționale $\mathcal{F}$.

Reamintim că elementele lui $\mathcal{F}$, simbolurile funcţionale, au ataşat un număr natural numit *aritate*. Cu $\mathcal{F}_n$ notăm mulţimea simbolurilor funcţionale de aritate n. Elementele lui $\mathcal{F}$ vor fi notate de obicei cu $f, g, h, f', g', f_1, g_2, h''$, etc. Elementele lui $\mathcal{F}_0$ se mai numesc simboluri *constante* şi se notează de obicei cu a, b, c, d, a_0, c_0, b', etc.

3.1 Alfabetul

Ca și formulele din logica propozițională, formulele din logica de ordinul I sunt șiruri de caractere peste un anumit alfabet. Spre deosebire de logica propozițională, alfabetul este mai bogat. Alfabetul logicii de ordinul I conține următoarele "caractere":

1. conectori logici deja cunoscuți: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \perp$, precum și doi *cuantificatori* noi: $\forall$, $\exists$;
2. variabile: vom presupune că avem la dispoziție o mulțime infinit numărabilă de variabile notată $\mathcal{X} = \{x, y, z, x', y', x_1, z'', \ldots\}$ (a nu se confunda cu variabilele propoziționale din logica propoziționale – sunt două noțiuni fundamental diferite);
3. simboluri auxiliare: "(", ")", "." și respectiv ",";
4. simboluri nelogice, care sunt specifice fiecărei signaturi în parte: simbolurile funcționale din mulțimea $\mathcal{F}$ și respectiv simbolurile predicate din mulțimea $\mathcal{P}$.

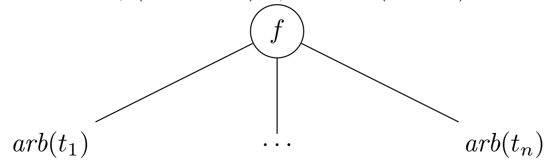
3.2 Termen

Definition 3.1. *Mulţimea* termenilor*, $\mathcal{T}$, este cea mai mică mulţime care satisface următoarele proprietăți:*

1. *$\mathcal{F}_0 \subseteq \mathcal{T}$ (orice simbol constant este termen);*
2. *$X \subseteq \mathcal{T}$ (orice variabilă este termen);*
3. *dacă $f \in \mathcal{F}_n$ (cu $n > 0$) şi $t_1, \ldots, t_n \in \mathcal{T}$, atunci $f(t_1, \ldots, t_n) \in \mathcal{T}$ (un simbol funcțional de aritate n aplicat unui număr de exact n termeni este termen).*

Termenii (sau, în mod echivalent, *termii*), sunt notaţi cu t, s, t_1, t_2, s_1, t', etc. Deşi termenii sunt scrişi în mod uzual ca un şir de caractere, ei au asociat un arbore abstract de sintaxă definit după cum urmează:

1. dacă $t = c$, $c \in \mathcal{F}_0$, atunci $arb(t) = c$
2. dacă $t = x$, $x \in \mathcal{X}$, atunci $arb(t) = x$
3. dacă $t = f(t_1, \ldots, t_n)$, $f \in \mathcal{F}_n$ $(n > 0)$, $t_1, \ldots, t_n \in \mathcal{T}$, atunci $arb(t) =$

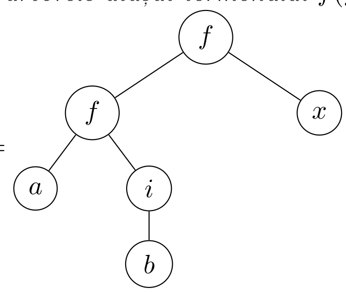


Example 3.1. *De exemplu, pentru signatura* $\Sigma = (P, Q, f, i, a, b)$ *definită mai devreme (*$ar(P) = ar(Q) = 2$*,* $ar(f) = 2$*,* $ar(i) = 1$*,* $ar(a) = ar(b) = 0$*), iată câteva exemple de termeni:* a*,* b*,* x*,* y*,* x_1*,* y'*,* $i(a)$*,* $i(x)$*,* $i(i(a))$*,* $i(i(x))$*,* $f(a, b)$*,* $i(f(a, b))$*,* $f(f(x, a), f(y, y))$*, etc.*

Practic, termenii se construiesc "aplicând" simboluri funcționale peste simboluri constante și variabile.

Example 3.2. *Deși formal termenii sunt definiți ca fiind șiruri de caractere peste alfabetul descris mai sus, aceștia trebuie înțeleși ca fiind arbori. De altfel, în orice software care lucrează cu termeni, aceaștia sunt memorați sub formă de arbori cu rădăcină. Iată arborele atașat termenului* $f(f(a, i(b)), x)$*:*

$arb(f(f(a, i(b)), x)) =$



3.3 Formule atomice

Definition 3.2 (Formulă atomică)**.** *O* formulă atomică *este orice șir de caractere de forma* $P(t_1, \ldots, t_n)$*, unde* $P \in \mathcal{P}_n$ *este un simbol predicativ de aritate* n*, iar* $t_1, \ldots, t_n \in \mathcal{T}$ *sunt termeni.*

Example 3.3. *Continuând exemplul anterior, folosim signatura* $\Sigma = (P, Q, f, i, a, b)$*, unde* $ar(P) = ar(Q) = 2$*,* $ar(f) = 2$*,* $ar(i) = 1$*,* $ar(a) = ar(b) = 0$*.*

Iată câteva exemple de formule atomice: $P(a, b)$*,* $P(x, y)$*,* $P\Big(f\big(f(a, i(x)), b\big), i(x)\Big)$*,* $Q(a, b)$*,* $Q\Big(i(i(x)), f(x, x)\Big)$*, etc.*

3.4 Formule de ordinul I

Definition 3.3 (Formule de ordinul I)**.** *Mulţimea formulelor de ordinul I, notată* `LP1` *este cea mai mică mulţime astfel încât:*

1. *(cazul de bază) orice formulă atomică este formulă (adică* $P(t_1, \ldots, t_n) \in$ `LP1` *pentru orice simbol predicativ* $P \in \mathcal{P}_n$ *și orice termeni* $t_1, \ldots, t_n$*; dacă* $n = 0$*, scriem* P *în loc de* $P()$*);*

2. *(cazurile inductive) pentru orice formule* $\varphi_1, \varphi_2 \in$ `LP1`*, pentru orice variabilă* $x \in \mathcal{X}$*, avem că:*

   (a) $\neg\varphi_1 \in$ `LP1`*;*

   (b) $(\varphi_1 \vee \varphi_2) \in$ `LP1`*;*

(c) $(\varphi_1 \land \varphi_2) \in$ *LP1;*

(d) $(\varphi_1 \to \varphi_2) \in$ *LP1;*

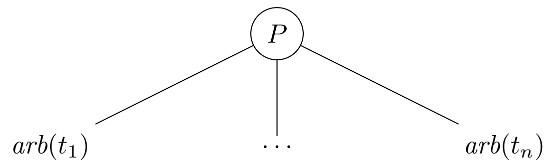
(e) $(\varphi_1 \leftrightarrow \varphi_2) \in$ *LP1;*

(f) $\forall x.\varphi \in$ *LP1;*

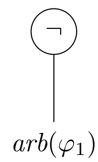
(g) $\exists x.\varphi \in$ *LP1.*

Remark 3.1. *Simbolurile predicate de aritate* 0 *țin locul variabilelor propoziționale (deocamtă, la nivel sintactic). Construcțiile* $\forall x.\varphi$ *și* $\exists x.\varphi$ *sunt noi.*

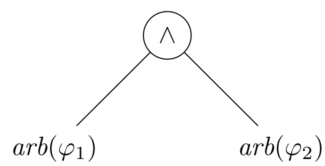
Formulele au asociat un arbore abstract de sintaxă definit în cele ce urmează:

1. dacă $\varphi = P(t_1, \ldots, t_n)$, atunci $arb(\varphi) =$ 

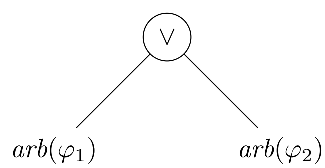
;

2. dacă $\varphi = \neg\varphi_1$, atunci $arb(\varphi) =$ 

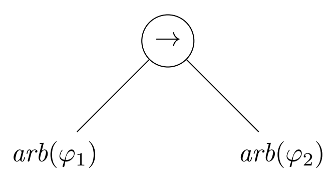
;

3. dacă $\varphi = (\varphi_1 \land \varphi_2)$, atunci $arb(\varphi) =$ 

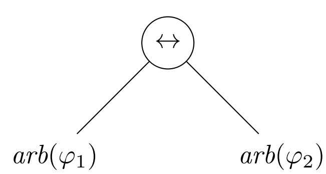
;

4. dacă $\varphi = (\varphi_1 \lor \varphi_2)$, atunci $arb(\varphi) =$ 

;

5. dacă $\varphi = (\varphi_1 \to \varphi_2)$, atunci $arb(\varphi) =$ 

;

6. dacă $\varphi = (\varphi_1 \leftrightarrow \varphi_2)$, atunci $arb(\varphi) =$ 

;

7. dacă $\varphi = (\forall x.\varphi_1)$, atunci $arb(\varphi) = \begin{array}{c} \forall x \\ | \\ arb(\varphi_1) \end{array}$;

8. dacă $\varphi = (\exists x.\varphi_1)$, atunci $arb(\varphi) = \begin{array}{c} \exists x \\ | \\ arb(\varphi_1) \end{array}$.

3.5 Paranteze

Parantezele sunt folosite pentru a marca ordinea efectuării operațiilor logice (și, sau, not, etc.) În continuare, vom renunța la parantezele de prisos, la fel ca în cazul logicii propoziționale: dacă o formulă poate fi interpretată ca un arbore abstract de sintaxă în două sau mai multe moduri, se vor folosi paranteze pentru a stabili arborele dorit.

De exemplu, formula $\varphi_1 \vee \varphi_2 \wedge \varphi_3$ ar putea fi înţeleasă ca $((\varphi_1 \vee \varphi_2) \wedge \varphi_3)$ sau ca $(\varphi_1 \vee (\varphi_2 \wedge \varphi_3))$. Pentru a nu polua formulele cu prea multe paranteze, se stabilesc *priorităţi*. În logică, ordinea priorităţii este: $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \forall, \exists$. În cazul în care nu suntem 100% siguri, este de preferat sa folosim paranteze suplimentare.

Din cauza priorității conectorilor logici, formula $\varphi = \varphi_1 \vee \varphi_2 \wedge \varphi_3$ va fi întotdeauna înţeleasă ca $(\varphi_1 \vee (\varphi_2 \wedge \varphi_3))$ (deoarece $\wedge$ este prioritar faţă de $\vee$). Ca analogie, la fel se întâmplă şi în cazul aritmeticii: $1 + 2 * 3$ va fi înteles ca $1 + (2 \times 3)$, deoarece $\times$ are prioritate în faţa lui $+$ ($\times$ este similar cu $\wedge$ şi $+$ cu $\vee$).

3.6 Exemplul 1

În continuare, vom explica care este signatura folosită pe a modela în logica de ordinul întâi afirmațiile *orice om este muritor*, *Socrate este om* și respectiv *Socrate este muritor*.

În primul rând, identificăm predicatele din text. Avem două predicate unare "este om" și respectiv "este muritor". Alegem simbolul predicativ *Om* pentru primul predicat și simbolul predicativ *Muritor* pentru al doilea predicat. De asemenea, în text avem și o constantă: Socrate. Alegem simbolul funcțional s de aritate 0 pentru această constantă. Așadar, pentru a modela afirmațiile de mai sus, vom lucra cu signatura

$$\Sigma = (Om, Muritor, s),$$

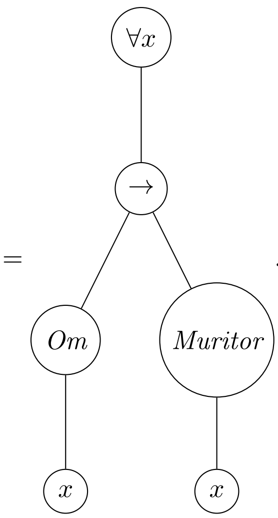
unde Om și $Muritor$ sunt simboluri predicative de aritate $ar(Om) = ar(Muritor) = 1$, iar s este un simbol constant de aritate $ar(s) = 0$.

Afirmația *orice om este muritor* va fi modelată prin formula de ordinul I

$$\forall x.(Om(x) \rightarrow Muritor(x)),$$

al cărei arbore abstract de sintaxă este:

$arb(\forall x.(Om(x) \rightarrow Muritor(x))) =$



.

Afirmația *Socrate este om* o vom modela prin formula atomică $Om(s)$, iar afirmația *Socrate este muritor* o vom modela prin formula atomică $Muritor(s)$.

Pentru signatura $\Sigma = (Om, Muritor, s)$ stabilită mai sus, există mai multe Σ-structuri posibile. Un exemplu de Σ-structură ar fi $S = (D, Om^S, Muritor^S, s^S)$ definită astfel:

1. D este mulțimea tuturor ființelor de pe Pământ;

2. $Om^S(x)$ este adevărat pentru orice ființă x care este și om;

3. $Muritor^S(x)$ este adevărat pentru orice ființă x (toate elementele domeniului sunt muritoare);

4. s^S este Socrate (Socrate, fiind o ființă, aparține mulțimii D).

Anticipând puțin (vom discuta despre semantica formulelor de ordinul I în cursul următor), toate cele trei formule discutate în această secțiune, $\forall x.(Om(x) \rightarrow Muritor(x))$, $Om(s)$ și respectiv $Muritor(s)$, sunt adevărate în structura S definită mai sus. De fapt, calitatea raționamentului *orice om este muritor; Socrate este om; deci: Socrate este muritor* este dată de faptul că formula $Muritor(s)$ este în mod necesar adevărată în *orice* structură în care formulele $Muritor(s)$ $\forall x.(Om(x) \rightarrow Muritor(x))$ și respectiv $Om(s)$ sunt adevărate, nu doar în structura S de mai sus.

3.7 Exemplul 2

Fie signatura $\Sigma = (<, =, +, -, 0, 1)$, unde $<$ și $=$ sunt simboluri predicative de aritate 2, $+$ este simbol funcțional de aritate 2, $-$ este simbol funcțional de aritate 1, iar 0 și 1 sunt simboluri constante.

Iată câteva formule care fac parte din limbajul de ordinul I asociat signaturii Σ:

1. $\forall x.\forall y.\Big(< (x, y) \rightarrow \exists z.(< (x, z) \wedge < (z, y))\Big)$;
2. $\forall x.\forall y.\exists z.\Big(= (+(x, y), z)\Big)$;
3. $\forall x.(< (0, x) \vee = (0, x))$;
4. $\forall x.\exists y.(= (x, -(y)))$;
5. $= (+(x, y), z)$.

De multe ori, în cazul simbolurilor predicative și simbolurilor functionale binare, se folosește notația infixată (e.g., $x < y$ în loc de $< (x, y)$). În acest caz, putem scrie formulele de mai sus în felul următor:

1. $\forall x.\forall y.\Big(x < y \rightarrow \exists z.(x < z \wedge z < y)\Big)$;
2. $\forall x.\forall y.\exists z.\Big(x + y = z\Big)$;
3. $\forall x.(0 < x \vee 0 = x)$;
4. $\forall x.\exists y.(x = -(y))$;
5. $x + y = z$.

Două dintre Σ-structurile posibile sunt $S_1 = (\mathbb{R}, <, =, +, -, 0, 1)$ și $S_2 = (\mathbb{Z}, <, =, +, -, 0, 1)$, unde predicatele și funcțiile sunt cele cunoscute de la matematică (cu precizarea că e vorba de funcția $-$ unar).

Anticipând cursul următor referitor la semantica formulelor de ordinul I, prima formulă este falsă în S_2 și adevărată în S_1. A doua formulă și a patra formulă sunt adevărate atât în S_1 cât și în S_2. A treia formula este falsă atât în S_1 cât și în S_2. Valoarea de adevăr a celei de-a cincea formule nu depinde doar de structura în care evaluăm formula, ci și de valorile variabilelor x, y, z. Deoarece variabilele x, y, z nu apar protejate de un cuantificator în formula numărul 5, acestea se numesc *libere*. Formula 5 este *satisfiabilă* atât în structura S_1 cât și în structura S_2, deoarece în ambele cazuri există valori pentru variabilele x, y, z care să facă formula adevărată (e.g. valorile $1, 2, 3$ pentru x, y și respectiv z).